HZ BOOKS
华章科技

淘宝资深软件开发工程师和系统运维工程师撰写，淘宝一线运维工作经验的总结，国内首部关于软件管理平台设计与实现的专著

不仅详细讲解了RPM和yum等软件管理工具的使用方法、技巧、原理和本质，还系统讲解了软件管理平台的规划、设计与具体实现

實戰

段继刚 著

The Development of Software Management Platform on Linux

Linux软件管理平台设计与实现

机械工业出版社
China Machine Press

The Development of Software Management Platform on Linux

Linux软件管理平台设计与实现

段继刚 著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Linux 软件管理平台设计与实现 / 段继刚著 . —北京：机械工业出版社，2013.9

ISBN 978-7-111-43792-5

Ⅰ. L… Ⅱ. 段… Ⅲ. Linux 操作系统 Ⅳ. TP316.89

中国版本图书馆 CIP 数据核字（2013）第 201758 号

本书是国内首部关于软件管理平台设计与实现（针对大规模 Linux 服务器集群）的著作，由淘宝资深软件开发工程师和系统运维工程师撰写，书中凝结了作者在淘宝运维一线积累的宝贵经验。不仅详细讲解了 RPM 和 yum 等软件管理工具的使用方法、技巧、原理和本质，还系统讲解了 Linux 软件管理平台的规划、设计与具体实现。对于从事 Linux 服务器管理和运维的工程师来说，本书将是不可多得的至宝。

全书共 8 章，在逻辑上分为三个部分：第一部分（第 1 ～ 4 章）为基础技术篇，详细介绍了 RPM 的概念、组织方式和制作方法，另外对 yum 服务进行了深入的分析，目的是为对软件开发感兴趣的读者在 RPM 和 yum 领域提供导引，为运维和系统管理人员提供指导，帮助读者在 RPM/yum 的使用和理解方面扫清障碍；第二部分（第 5 ～ 6 章）为问题分析和系统设计篇，结合企业中软件管理的现状，提出问题，然后从基本的系统开始，逐步解决问题，优化方案，最终形成一套软件管理平台的设计方案，适合在企业中负责软件平台优化和系统服务管理的人员阅读；第三部分（第 7 ～ 8 章）为实现篇，如果读者想将本书中设计的软件管理平台应用到自己所在企业的环境中，可以参考这部分的内容进行详细设计和编码实现。附录列举了 RPM 和 yum 在日常使用中经常会遇到的问题以及对应的解决方法。

机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码 100037）

责任编辑：孙海亮

北京市荣盛彩色印刷有限公司印刷

2013 年 9 月第 1 版第 1 次印刷

186mm × 240mm • 14.25 印张

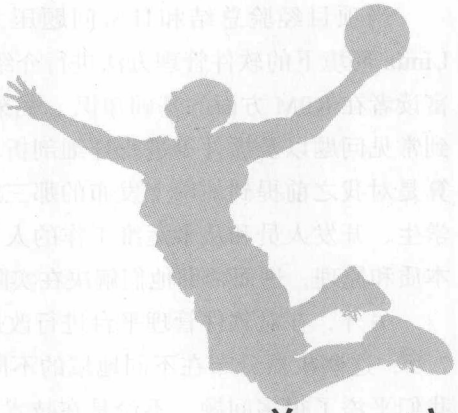
标准书号：ISBN 978-7-111-43792-5

定 价：59.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：（010）88378991 88361066 投稿热线：（010）88379604

购书热线：（010）68326294 88379649 68995259 读者信箱：hzjsj@hzbook.com



前　言

为什么要写这本书

在2011年以前，我一直从事软件开发工作，后来开始参与运维工作，一半运维，一半开发。我在将自己开发的程序部署到公司几万台服务器上的过程中遇到了一些RPM和yum使用上的问题，于是便花了很长时间对RPM和yum等软件管理工具进行了学习，对RPM的格式剖析、RPM的制作以及yum的使用、原理和详细配置等进行了总结，并将收获写成文章发布在网上，希望能帮到有同样需求的同行们。文章发布后不久，意外发现，对Linux环境下的RPM和yum感兴趣的同行竟然有这么多。因此，便计划对这块知识进一步学习，并进行整理，以期能帮到更多朋友。

碰巧在2012年，公司的三个子公司合并，各个子公司负责yum服务管理的同事由于离职或者转岗等原因，不再从事原来的yum维护工作，此事便交给我来负责。后来，部门又成立了软件配置统一规范化管理的项目，我也顺理成章地被安排来负责该项目。在经过了大半年的整合和改造后，公司的软件管理服务终于能够基本统一，使用同一套服务、同一套仓库、同一套配置，这样，Linux软件包的管理工作就变得轻松起来了（事实上，我们还有很多事情要做）。

2013年春节过后，由于个人岗位调整的原因，原来负责的软件管理平台整合和改造工作要交给别的部门来维护，为了不让学习到的知识和经验废弃，我便计划对这个项目中积累的知识和经验进行总结。而且在2012年这一年的时间中，几乎每天都会被同事的RPM和yum的使用问题“骚扰”，我和同事们也一直在尝试通过某种途径来减少这些问题的发生，比如通过培训分享的方式来对RPM、yum和公司的软件管理服务进行介绍，以提高部门中每个人解决RPM和yum问题的能力，从而减少系统管理员的工作量，提高部门工作效率。

当项目经验总结和日常问题压力这两个理由汇总到一起时，我便决定写一本书，对Linux环境下的软件管理方法进行介绍。首先，对RPM文件及SPEC文件等进行分析，以丰富读者在RPM方面的基础知识；另外，对RPM管理工具——yum，从原理到使用细节，再到常见问题以及插件等进行详细剖析，这部分可以说是Linux软件包管理的中高级话题，也算是对我之前提到的网上发布的那三篇文章的细化和扩展。我期望对RPM和yum感兴趣的学生、开发人员和从事运维工作的人员通过对这部分技术的学习，能够了解RPM和yum的本质和原理，进而帮助他们解决在实际工作中使用这些工具时遇到的大多数问题。

另外，在对软件管理平台进行改造时，我们的软件管理平台支持的服务器数量已经接近7万，这些机器分布在不同地域的不同城市中，处在不同的网络环境中，这个复杂的情况为我们平添了很多问题。不论是在技术方面，还是业务方面，我们都花费了不少时间和精力，最终才形成了一个能够支撑整个公司服务器集群的软件管理平台。可以说，这个软件管理平台和伴随这个项目总结的经验都来之不易，而且是弥足珍贵的，因此我决定通过书籍的方式来将这些经验和方案分享给广大的同行。

在上述的软件管理平台改造过程中，我通过和其他公司的朋友进行沟通了解到，在一些较大的软件或者互联网公司，软件管理平台仍然是一个很简易和初级的系统，大家对软件管理规范的概念都很淡漠。所以在本书的后半部分，我为同样从事软件管理工作的同行提供了一套方案，也是一份礼物，即从软件管理平台的最初模型开始，对我在实际工作中的经验和遇到的问题逐个进行剖析，在这个过程中总结经验、不断改造，最终形成了一套软件管理平台的设计方案。希望这个方案能够给同行企业或者感兴趣的读者提供参考。总之，我希望本书的内容能够帮助读者提高工作效率，当然更希望它能帮助同行的公司增加收益。

另外，关于Linux环境中软件管理平台设计方法的书籍，在市场上还鲜有出现，因此，笔者希望本书的出版，能够在图书领域和技术领域，都打开一个新的篇章，让我们的图书出版行业多一个软件管理平台的类目；也让我们的技术领域，多一个软件管理平台的研究方向。

以上就是本书撰写的初衷。

读者对象

本书的读者对象如下：

- 对Linux环境下软件开发感兴趣的技术人员
- 从事计算机系统管理的工作人员
- 互联网公司技术部的运维人员
- 对软件架构设计和优化感兴趣的工程人员
- 计算机相关专业的高等院校学生

如何阅读本书

本书共分 8 章，按照讲述的内容和适应读者的不同层次，可以划分为三大部分：

第一部分为基础技术篇，包括第 1 ～ 4 章，介绍了 RPM 的概念、组织方式和制作方法，另外对 yum 服务进行了深入的分析。这部分的目的是为对软件开发感兴趣的读者在 RPM 和 yum 领域提供导引，为运维和系统管理人员提供指导，帮助这部分读者尽可能扫清在 RPM 和 yum 的使用和理解方面的疑问。

第二部分为问题分析和系统设计篇，包括第 5 章和第 6 章，结合企业中软件管理的现状，提出问题，然后从基本的系统开始，逐步解决问题、优化方案，最终形成一套软件管理平台的设计方案。这部分内容适合在企业中负责软件平台优化和系统服务管理的人员阅读。在阅读这部分内容时，从事运维 / 系统管理的读者大多会找到能与笔者产生的共鸣之处，或者会有一种“身临其境”、“似曾相识”的感觉，因为这部分所涉及的索引更新问题、多机房镜像问题、数据同步问题以及 RPM 包冲突问题等，都是 Linux 软件管理工作中最容易遇到的问题，而我们的讨论正是针对这些常见问题展开的。

第三部分为实现篇，包括第 7 章和第 8 章。如果读者想将本书中设计的软件管理平台应用到自己所在企业的环境中，可以参考这部分的内容进行详细设计和编码实现。

附录列举了 RPM 和 yum 在日常使用中经常会遇到的问题以及对应的解决方法。

勘误和支持

由于作者的水平有限，加之编写时间仓促，书中难免会出现一些错误或者不准确的地方，恳请读者批评指正。如果大家有任何宝贵意见，可发送邮件到 cmesoft@126.com 或者 duanjigang1983@gmail.com，期待您的真挚反馈。

另外，书中提到的示例代码和最终软件管理平台的部分代码，在 https://github.com/duanjigang1983/lsmd 上会提供下载和后续更新，欢迎读者朋友们跟进，并与我交流。您也可以到华章公司的网站下载相应的源代码，华章公司网站的网址为：http://www.hzbook.com。

致谢

首先要感谢 RPM 和 yum 的开发者，没有他们的付出，就不会有 RPM 和 yum 这样优秀的工具产生，Linux 系统下的包管理工作也不会像现在这样便利。

在本书定稿的前一天，也就是 2013 年 7 月 10 日，yum 工具的开发者 Seth Vidal 先生因为车祸不幸去世，我们为计算机领域失去这位专家感到惋惜。虽然本书的内容远不能达到 Seth Vidal 先生对计算机领域所做贡献的高度，但是我仍希望本书能够和 RPM、yum 工具一起，在被读者阅读和使用时，能使他们想起曾经有一位伟大的开发者——Seth Vidal，为计算

机行业做出了很大的贡献。

感谢参考文献中提及的所有文章或书籍的作者，正是有了这些作者无私的分享，我才能对 RPM 和 yum 技术进行全面的学习和研究。

还要感谢我所在公司的诸多同事：感谢于霆在运维方面对我的指导；感谢王波和杨家宁在系统管理方面对我的影响；感谢谢杰灵、李拓和刘轶在智能 DNS 技术方面为我提供的帮助；感谢赵立文和司加详在网络技术方面对我提供的支持；感谢陆严、王惠军和张雅群等在工作中对我的支持；感谢周琪凯等对本书出版的关注；感谢周末篮球小组中的李一鑫、杨旭、林杰和戴海军四位同事对本书出版的支持。

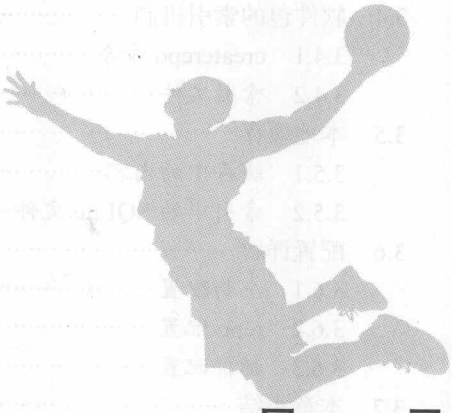
另外还要感谢 ChinaUnix 社区的周荣茂、白金、刘岐、王东洋、陈小玉、江均勇、方云麟和张飞对本书出版的支持。

感谢机械工业出版社华章公司的编辑杨福川老师和孙海亮老师在书籍编写过程中对我的帮助。

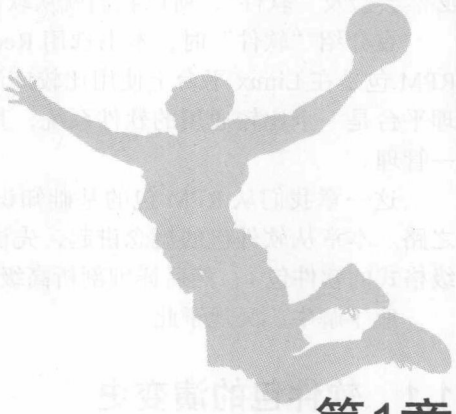
最后感谢我的家人对我的支持，感谢我的女朋友在生活中对我的关怀、包容和支持！

段继刚

2013 年于杭州西溪



目　录



第1章
RPM 认知与格式剖析

- 软件包的演变史
- RPM 软件包基础知识
- RPM 格式剖析
- RPM 解析例程
- 本章小结

本书主要讲述 Linux 环境下的软件管理平台是如何设计和实现的。要说明这个问题，就必然会涉及“软件”，所以我们就从软件谈起。

在介绍“软件”时，本书选用 RedHat 系统上的 RPM 格式的软件包作为讨论对象，因为 RPM 包是在 Linux 平台上使用比较广泛的一种包格式。另外，我们将要设计和实现的软件管理平台是一个比较通用的软件系统，其他格式的软件包也可以参照本书中提供的方法进行统一管理。

这一章我们从 RPM 包的基础知识介绍起步，带领读者踏入软件管理平台的设计和实现之路。本章从软件包的概念讲起，先讲述软件包进化过程（低级格式的软件包怎样发展到高级格式的软件包）；然后详细剖析高级格式软件包的典型案例——RPM。

庖丁解牛，又现于此。

1.1 软件包的演变史

关于软件包，在不同的时期，具有不同的形态。最早期的软件包是一些可以运行的程序组成的集合，可能还要加上若干配置文件和动态库。举个例子，程序员把自己写好的脚本文件或者二进制文件（源码编译生成的）、程序所依赖的动态库文件（比如以 .so 和 .dll 为扩展名的文件）及配置文件复制到一个目录中，取名为 execute，这时 execute 就可以称为一个软件包了。

有读者可能会有疑问：这样就能称为软件包了？是的，它确实是一个软件包，因为这个目录当中已经包含了一个应用运行需要的所有东西：从可执行程序到配置文件，再到共享库等，都具备了。之所以有人怀疑其能否称得上是软件包，是因为它看起来有些原始和简单罢了。上面所描述的软件包，在短期内能够满足一些简单应用的需求。然而，随着应用的不断复杂化和技术的不断革新，这种格式的软件包逐渐不能满足人们的需要了。比如，某个工程的软件包包含了数百个文件，每次把这些文件复制并安装到别的机器上，都要花费不少时间。另外，当某个软件包中的文件被意外删除或者修改后，管理人员并不知道，这样就会导致在别的环境下，虽然软件能够正常安装，但是在运行时，可执行程序很可能不能正常工作。于是，人们开始规划更高级别的软件包。

为了保证使用的软件包能够方便且快速地复制到别的机器上，人们开始选用压缩文件的方式来封装软件包。比如对 execute 进行压缩后，它就成了另外的一种软件包：通过 tar 或者 gzip 压缩后得到 .tar.gz、.rar 或者 .zip 格式的文件，这时我们就获得了一个较为高级的软件包。之所以称其为高级软件包，是因为这种格式的软件包把程序和配置压缩成了一个单一的文件，这样的格式既方便数据复制，又节省了磁盘空间，在通过网络传输时，还能减少带宽资源。另外，为了保证每次的软件修改都能被安装者区分，一些制作压缩格式软件包的开发人员会用一个校验和去标识新发布的软件包，比如通过 md5sum 校验和来供使用者核对包的完整性。我们在很多下载站点经常看到这样的现象：在页面中资源下载链接的后面，通常会紧跟着一个 md5sum 的字符串，当使用者完成软件包下载后，就可以在他的机器上通过以下命令：

```
md5sum 文件名
```

来计算获取软件包的 md5sum 校验和，然后和下载页面的 md5sum 值进行对比：如果两者一致，说明下载的包是完整的；如果不一致，说明下载的包有问题。通常，大多数开发者也会为压缩包加上一个自定义的版本信息，比如，test-1.1.tgz 文件就是一个针对 test-1.0.tgz 软件包的升级版安装包。

笔者曾经参与过一个持续了两年的项目，项目组的程序基本都是通过压缩包的方式来发布的。每次发布程序时，维护人员都必须通过手动的方式，把动态库、二进制程序和配置文件压缩成一个 .tgz 文件，复制到要部署的 2000 多台机器上，并解压缩到指定目录下，然后直接启动运行。另外，经常光顾 kernel.org 的读者应该也留意到了，在内核源码的下载页面中，通常的下载资源都是采用压缩包加版本的格式来提供下载的，而且伴随每个下载链接，通常都会有一个 .sign 文件，称为特征文件，这个文件中存储了下载文件的 GPG 签名（参考 GPG 相关文档），这也是一种典型的压缩格式的软件包。我们可以看到，在日常使用方面，相对于最原始的零散文件方式的软件包，压缩格式的软件包已经比较方便了。

再往后发展，就出现了更高级的软件包，比如 .rpm、.bin 或者 .deb 格式的软件包。这些格式的软件包，相对于压缩格式的软件包又有了更进一步的发展，它们不仅支持文件压缩功能，还有依赖维护、脚本的嵌入等功能。要了解每种软件包的具体功能，需要读者去翻阅对应的文档。RedHat 公司开发贡献的 RedHat Package Manager（RPM）可以说是这些高级别软件包中最典型的一个，在本书中，我们将会选取这种格式的软件包作为讨论对象。

1.2 RPM 软件包基础知识

有了 .tgz、.rar 或者 .zip 格式的软件包，日常学习和工作就够用了吗？对于某些小型应用来说，这样确实能满足需求了。然而，在工作中往往会碰到如下一些需求，使得我们不能满足于停留在这个层面的软件包上：

- 查看某个软件包的安装时间、制作人、制作时间、描述信息等。
- 使软件包具备一些特殊功能，比如，除了文件复制外，还要能生成配置文件，并且能提供安装服务、执行命令等。
- 当软件版本升级时，通过压缩文件重新覆盖的做法在文件很大时显然是不可行的，此时就会希望使数据量更新达到最小。

除了以上 3 点，可能还有其他方面的问题，相信读者或多或少都遇到并思考过。那么，是否存在某个格式的软件包，这种软件包能在文件压缩存储之外再支持更多的功能呢？答案是肯定的，这就是我们要介绍的 RPM 软件包。

1.2.1 RPM 软件包的功能

RPM 就是能够解决上面提到的问题的若干种软件包中的一类。当然还有其他格式的软件包也能解决这些问题，不过本书只对 RedHat 系统上的 RPM 进行讨论。

RPM 软件包的功能如下：

- 存储数据压缩。
- 文件安装。
- 配置文件生成。
- 系统服务注册。
- 软件依赖检查。

除了前两个功能外，其他 3 个功能是一般的压缩格式软件包都不具备的。下面详细介绍 RPM 的这几个功能。

1. 存储数据压缩

RPM 具有软件包的基本功能——数据压缩存储，它支持常见的 bz2 和 gzip 等压缩算法。RPM 安装列表中的所有文件在按照某个指定的算法（默认是 gzip）压缩后，作为最终 RPM 文件的一个数据块，与其他控制信息存储进同一个文件中。最终所有的数据都存储在同一个 RPM 文件中。

读者可以在制作 RPM 前先计算一下安装包的大小，生成 RPM 后，再计算一下 RPM 文件的大小，两者一比较就能得到 RPM 的压缩效果了。比如笔者开发的一个测试包，安装目录大小为 5.2MB，而该目录生成的 RPM 文件却只有 2.4MB。

2. 文件安装

文件安装是软件包的一个基本功能，比如我们运行如下命令：

```
rpm -qpl ./cmeguard-1.1.2-34.i386.rpm
```

输出内容如下：

```
/etc/init.d/cmeguard
/usr/local/cmeguard/bin/cmesync
/usr/local/cmeguard/bin/daemon
/usr/local/cmeguard/bin/genfinger
/usr/local/cmeguard/bin/run
/usr/local/cmeguard/bin/sync_plug
/usr/local/cmeguard/conf/cmeguard.conf
```

这里输出的所有文件，最终都会在软件包安装时被安装到目标操作系统上，在使用以下默认选项安装这个包之后，就可以在系统上找到以上列表中所有的文件了。

```
rpm -ivh cmeguard-1.1.2-34.i386.rpm
```

一般情况下，被安装的文件不仅仅是这里能看到的文件（事实上，更多的文件可能被安装了），因为 RPM 除了会安装文件列表中指定的文件外，还会在安装过程中动态生成一些文件，并且安装到系统中。具体介绍参见后面章节。

3. 配置文件生成

配置文件既可以指定为安装列表中的文件，也可以通过安装过程中运行的脚本来生成。如果是静态文件，直接加入安装目录中，通过文件列表的方式来安装；如果配置文件的内容和所安装的系统的运行时信息相关，那么就需要在安装过程中来动态生成了。例如，本书后面提供的名为 repoutils 的软件包，在生成以下配置文件时，就是根据操作系统的版本信息和

平台信息来动态生成文件中的 baseurl 项的内容的。

```
/etc/yum.repos.d/redhat.repo
```

4. 系统服务注册

如果你曾使用 rpm 命令安装过一些软件包，比如常见的 apache、mysql-server 等，应该会知道，在安装完成后，目录 /etc/init.d/ 下会生成一个 httpd 或者 mysqld 文件。这个文件是按照标准的 Linux 服务脚本的格式编写的（参考 chkconfig 的文档）。当系统以对应的模式启动时（参考 Linux run levels 文档），之前安装的服务程序就会被启动运行，因为 /etc/init.d 目录下的每一个文件都会被系统识别为一个服务控制命令，系统在启动过程中，会向每个安装命令（比如 httpd）传递一个 start 参数来执行该命令，从而触发该服务（httpd）的启动，关机过程中对服务的停止也是类似的。

举个例子，运行以下命令：

```
rpm  -qpl ./vsftpd-2.0.5-10.el5.i386.rpm  | grep init
```

输出如下：

```
/etc/rc.d/init.d/vsftpd
```

说明 RPM 包中的这个文件默认会被安装到系统的如下位置。

```
/etc/rc.d/init.d/vsftpd
```

安装完成后，可以通过以下命令来启动 vsftpd 服务。

```
service vsftpd start
```

通过以下命令来停止 vsftpd 服务。

```
service vsftpd stop
```

通过以下命令来重启 vsftpd 服务。

```
service vsftpd restart
```

或者通过以下命令来查看这个服务的运行状态。

```
service vsftpd status
```

5. 软件依赖检查

在工作中，我们开发的程序很少单独就能运行，大多数程序都会依赖其他组件。比如，数据库操作程序可能需要 libmysql 的支持，网络报文处理程序或许需要 libpcap 库的支持。

为了保证每个软件在安装后都能正常运行，在安装之前或者安装过程中，软件安装程序需要对该软件包所依赖的所有元素进行检查。而这种检查机制，就可以通过 RPM 的 require 等语法来实现（参见第 2 章关于 SPEC 文件的内容）。例如，有个 RPM 文件，其文件名称为 test_rpm-1.1.1-21.i386.rpm，查看该包所依赖（require）的服务或者组件的命令如下：

```
rpm -qp test_rpm-1.1.1-21.i386.rpm -requires
```

输出如下：

```
  test__require_pkg
ruby-libs
    /bin/sh
    /bin/sh
rpmlib(PayloadFilesHavePrefix) <= 4.0-1
rpmlib(CompressedFileNames) <= 3.0.4-1
```

上面的输出内容就是 test_pm 这个包在安装时所依赖的所有元素。

然后尝试如下安装命令：

```
rpm -ivh test_rpm-1.1.1-21.i386.rpm
```

能够看到如下输出：

```
error: Failed dependencies:
test__require_pkg is needed by test_rpm-1.1.1-21.i386
```

我们发现，由于系统缺少 test__require_pkg 这个软件包（或者虚拟包），从而导致 test_rpm-1.1.1-21.i386.rpm 的安装失败。另外，执行如下命令来查看 ruby-libs 这个包的安装情况：

```
rpm -qa ruby-libs
```

输出如下：

```
ruby-libs-1.8.5-5.el5_3.7
```

从中能够看到，虽然 test-rpm 所依赖的 ruby-libs 软件包已经安装了，但是 test-rpm 还是不能安装，这是因为，在一个软件包的依赖列表中（test__require_pkg 和 ruby-libs），所有依赖组件都是“与”的关系，也就是说，缺少任何一个依赖组件（test__require_pkg），目标 RPM 都不能安装。

1.2.2 RPM 实现引子

为了引出后面章节的内容，我们在这里对 1.2 节开始部分提的问题进行简单解答。

RPM 是怎样在压缩存储之外又实现其他功能的呢？比如：安装服务、执行命令、打印信息、发邮件、检查依赖包，还有升级时做版本检查等功能。

关于这些功能的实现，在此只做简单介绍，细节留待有关 SPEC 的章节再作说明。

阅读后面的章节我们会了解到，RPM 有个功能就是对嵌入脚本的支持：它支持在安装软件或者卸载软件的过程中（确切地说，是这个过程的开始、进行时和结束后这几个不同的动作点），执行用户指定的命令。常用的命令执行点如下：

- pre install
- post install
- pre uninstall
- post uninstall

从字面就能看到每个执行点的含义：pre/post install 表示在安装之前或者之后；pre/post uninstall 表示在卸载之前或者之后。

这种支持执行自定义嵌入脚本的机制，为扩展软件包的功能提供了很大的空间。

再来看一个例子，执行如下命令来查看指定 RPM 中嵌入的脚本：

```
rpm -qp test_rpm-1.1.1-21.i386.rpm -scripts
```

输出如下：

```
preinstall scriptlet (using /bin/sh) :
echo "pre install scripts by duanjigang"
postinstall scriptlet (using /bin/sh) :
echo "post install scripts by duanjigang"
preuninstall scriptlet (using /bin/sh) :
#!/bin/bash
echo "pre uninstall by duanjigang"
postuninstall scriptlet (using /bin/sh) :
#!/bin/bash
echo "post uninstall by duanjigang"
```

从输出情况能够看到，这个软件包嵌入了 4 个脚本，分别在安装前后和卸载前后执行，这样，使用者就可以通过这些脚本来实现自己想要的功能了。

关于 scripts（脚本）在哪些动作点执行时会有什么效果，在后面会详细说明。

另外一个问题：RPM 是怎样实现版本升级的？

执行如下命令查看 RPM 包的信息：

```
rpm -qpi test_rpm-1.1.1-21.i386.rpm
```

可以查看到该 RPM 包的信息如下：

```
Name        : test_rpm
Version     : 1.1.1
Release     : 21
```

我们看到，test-rpm 的 Version 是 1.1.1，Release 是 21，因此它默认的包名如下：

```
test_rpm-1.1.1-21
```

在对 version 或 release 进行升级后，新生成的 RPM 就可以用来升级老版本的安装包了。rpm 命令会自动检查待安装包和已安装软件包的信息，确认待安装包是否能升级已安装软件包。软件的升级是一个复杂的过程，必须谨慎地考虑每一个步骤，保证升级时所做的一切操作尽可能少地影响已经安装软件的正常运行，只有这样才能完成新版本软件的安装。

RPM 的基础知识就先介绍到这里。

1.3 RPM 格式剖析

在对 RPM 格式的软件包有了基本的认识之后，本节将以开发人员的身份，从协议的角度来剖析 RPM 的组成结构。

1.3.1 从协议说起

如果你做过 TCP/IP 应用相关程序开发或者熟悉 wireshark（或者 tcpdump）等工具，应该